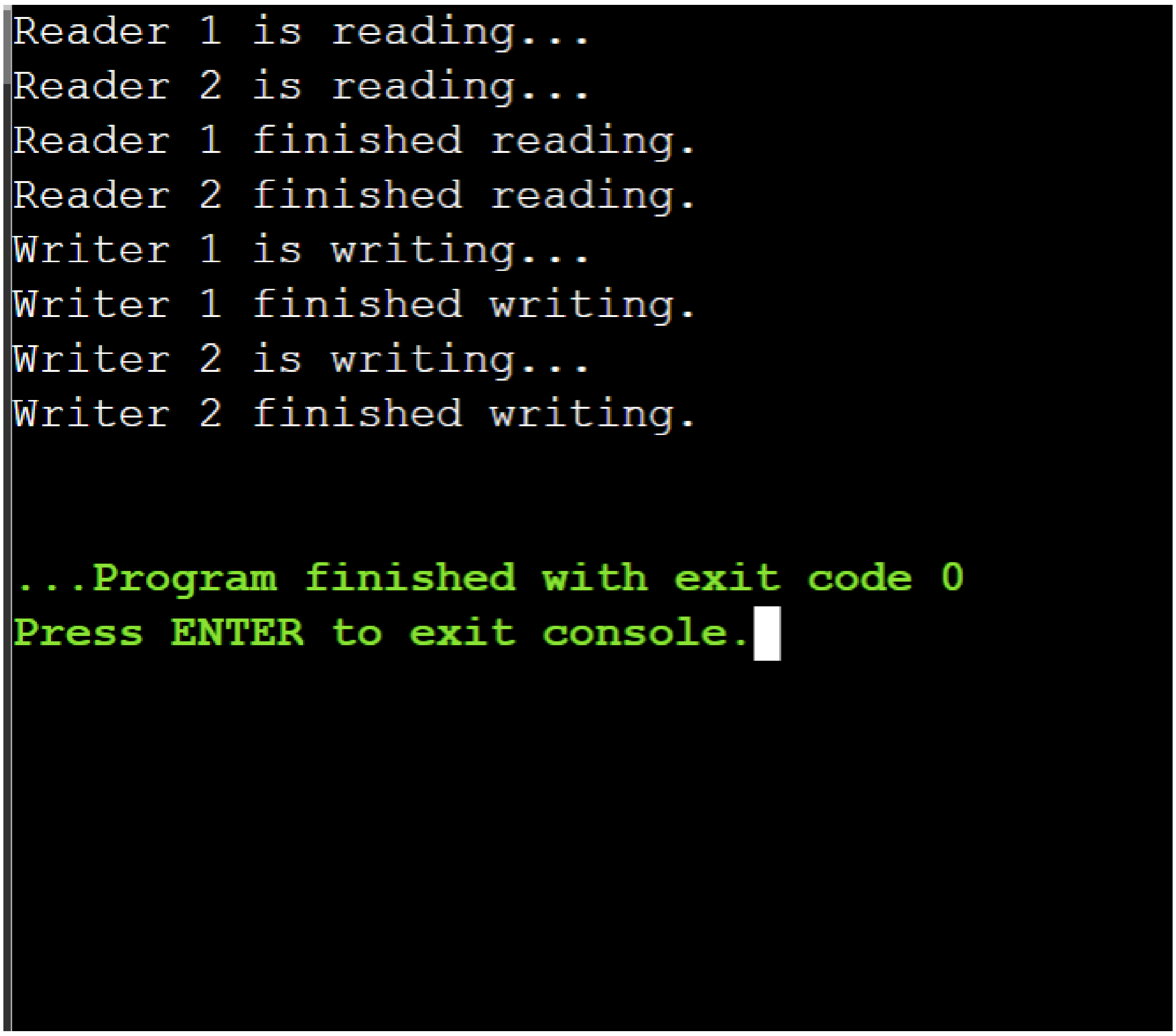
```c
#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>


sem_t mutex;        // Protects readCount

sem_t writeLock;     // Allows one writer or multiple readers

int readCount = 0;   // Number of active readers


void* reader(void* arg) {

    int id = *(int*)arg;


    sem_wait(&mutex);

    readCount++;

    if (readCount == 1) {

        sem_wait(&writeLock);  // First reader locks the writers out

    }

    sem_post(&mutex);


    // Critical section (reading)

    printf("Reader %d is reading...\n", id);

    sleep(1);

    printf("Reader %d finished reading.\n", id);


    sem_wait(&mutex);

    readCount--;

    if (readCount == 0) {

        sem_post(&writeLock);  // Last reader unlocks the writers
    }
    sem_post(&mutex);

    return NULL;
}

void* writer(void* arg) {
    int id = *(int*)arg;

    sem_wait(&writeLock);  // Writer locks access
    // Critical section (writing)
    printf("Writer %d is writing...\n", id);
    sleep(2);
    printf("Writer %d finished writing.\n", id);
    sem_post(&writeLock);  // Release lock

    return NULL;
}

int main() {
    pthread_t r1, r2, w1, w2;
    int id1 = 1, id2 = 2;

    // Initialize semaphores
    sem_init(&mutex, 0, 1);
    sem_init(&writeLock, 0, 1);

    // Create threads
    pthread_create(&r1, NULL, reader, &id1);

    pthread_create(&w1, NULL, writer, &id1);

    pthread_create(&r2, NULL, reader, &id2);

    pthread_create(&w2, NULL, writer, &id2);


    // Wait for threads to finish

    pthread_join(r1, NULL);

    pthread_join(w1, NULL);

    pthread_join(r2, NULL);

    pthread_join(w2, NULL);


    // Destroy semaphores

    sem_destroy(&mutex);

    sem_destroy(&writeLock);


    return 0;

}
```



```
Reader 1 is reading...
Reader 2 is reading...
Reader 1 finished reading.
Reader 2 finished reading.
Writer 1 is writing...
Writer 1 finished writing.
Writer 2 is writing...
Writer 2 finished writing.


...Program finished with exit code 0
Press ENTER to exit console.
```